IN THE SPECIFICATION

Please amend the first paragraph on page 1, lines 4-5, as shown below.

--The present invention relates to a computer system with ~~localised~~ localized on-chip debug facility.--

Please amend paragraphs 11 and 12 on page 5, lines 24-27 as shown below.

-- FIG. 9 illustrates schematically FIFOs in the ~~synchronisation~~ synchronization unit; and

FIG. 10 illustrates a timing diagram for operation of the ~~synchronisation~~ synchronization unit.--

Please amend the first full paragraph on page 6, lines 8-20, as shown below.

--Figure 2 is a more detailed schematic of the processor 4 in combination with selected on-chip emulation functional blocks which form part of the on-chip emulator 6. The processor 4 comprises a processor core 18 which is connected to a program memory 20 which holds code to be executed by the core. The core comprises a prefetch/align stage 22, a decode/dispatch stage 24, a microinstruction generator 26 and four parallel execution pipelines AU_0, AU_1, DU_0, and DU_1. The ~~core~~ processor 4 operates in a pipelined manner such that all stages can be active at the same time, on different instructions. The pipeline stages are denoted by horizontal dotted lines in Figure 2. It will readily be understood that each execution pipeline itself AU_0, AU_1, DU_0, and DU_1 constitutes a number of pipeline stages.—

Please amend the first two paragraph on page 7, lines 1-9, as shown below.

--The program memory 20 contains code in the form of ~~128-bit~~ 128-bit long words. Each word contains a plurality of instructions depending on the instruction mode of the processor. In this respect, reference is made to Figure 3.

According to a first instruction mode, a pair of ~~16-bit~~ 16-bit instructions are supplied during each machine cycle to the decoder 24 from the prefetch/align stage buffer 22. ~~this~~ This

pair is denoted slot0, slot1 in bit sequences w0, w1 etc. This is referred to herein as GP16 superscalar mode.—

Please amend the fifth paragraph on page 7, lines 18-23, as shown below.

--In all modes, each fetch operation initiated to the program memory 2 retrieves an instruction word of 128 bits in length. Thus, in GP16 mode, the instruction word comprises eight ~~16-bit~~ 16-bit instructions, paired as slot0, slot1 for each machine cycle. In GP32 and VLIW mode, the instruction word comprises four ~~32-bit~~ 32-bit instructions.—

Please amend the first full paragraph on page 9, lines 10-29, as shown below.

--The OCE block 6 comprises a PC watch block 32 which snoops addresses issued by the prefetch align state 22 to the program memory 20. Addresses which it is programmed to match are held in a watch register 34. When the PC watch block 32 sees an address which it has been programmed to watch for, so-called diagnostic flags ~~diag~~ DIAG are added to the instruction word before it is supplied to the prefetch/align stage 22. There are eight possible such diagnostic flags. The diagnostic flags are set according to the different instruction modes. If a diagnostic flag is set this denotes that a particular instruction is a PC (program count) watch, i.e., an instruction of interest to the emulator 6. Figure 4 illustrates how the diagnostic flags are set in GP16 and GP32 modes. There are eight diagnostic flags because there can be up to eight GP16 instructions in a 128 bit fetch line. The PC watch unit has an exact address of each PC it is looking for and sets the corresponding flag according to the positions illustrated in Figure 4. In GP16 mode, each flag can be set. In GP32 mode, diagnostic flags can be set on bits 0,2,4 and 6 denoting each GP32 instruction respectively. In VLIW mode, four GP32 diagnostic flags (bits 0,2,4 and 6) can be set.--

Please amend the paragraph beginning on page 11, line 12 through page 12, line 4 as shown below.

-- In order to implement precise and non-precise PC watching, the on-chip emulator 6 has an OCE control unit 36 and a ~~synchronisation~~ synchronization unit 38 for ~~synchronising~~

synchronizing the program count of instructions under watch with commit values supplied by the data unit. The decode/dispatch stage 24 has its own program counter 40 which holds the PC values of instructions currently being decoded. The decode/dispatch unit 24 supplies the PC of each instruction which it receives, together with a flag identifying PC watch instructions, to the OCE control unit 36 along line 42. The PC values are held in a PC FIFO in the ~~synchronisation~~ synchronization unit 38. Commit values from the data unit are held in a commit FIFO in the ~~synchronisation~~ synchronization unit, and the OCE only validates the PC watch once it sees that the instruction at that PC was committed. The action of the emulator 6 while awaiting guard resolution depends on whether it is in precise or non-precise watch mode. A signal is returned to the decode/dispatch unit 24 on line 44 indicating the status of the watch. The OCE control block 36 can also set the processor into a debug mode. To achieve this, an architectural bit PSR.DIAG is set inside the program status register 47. This bit is sent with the fetch address to a program memory controller 21. If clear, the fetch is made from the normal program memory. This bit controls a multiplexor 48 in the program memory controller. The on-chip emulator 6 has its own program memory 50 which holds debug code which is executed by the processor core 18 when in debug mode. When the PSR.DIAG bit is set, the fetch is made from the OCE program memory 50 instead of from the normal program memory 20. Thus, fetched debug instructions are executed by the processor as normal instructions.--

On page 12, please amend the second full paragraph, lines 14-27, as shown below.

--At the microinstruction generator 26, guards to be resolved are sent to the data unit pipelines DU_0, DU_1 in the same microinstruction that the current machine instruction has been encoded as. The manner in which this is done is described later. If the guard is resolved such that the instruction is executed, a Commit signal 52 is sent from the data unit pipelines DU_0, DU_1 to the on-chip emulator 6. The on-chip emulator 6 checks the PC value in the PC register ~~38~~ 28 and determines from the watch register 34 what action should be taken in the event of a committed instruction at that PC value. If that action is a divert, a divert signal 46 is issued to the multiplexor 48, and the prefetch/align stage is issued with a branch address to debug code so that the processor can from that point execute debug code from the OCE program memory 50.—

Please amend the paragraph beginning on page 16, line 31 through page 17, line 7, as shown below.

-- It will be appreciated that at any one time the on-chip emulator 6 may hold a number of PCs indicating PC watch instructions which are waiting for respective Commit signals to be returned from the data unit pipelines. This is dealt with in the on-chip emulator by the ~~synchronisation~~ synchronization unit 38 which includes incoming PCs and Commit values. This allows the Commit signals to be associated with the correct PCs. This is discussed in more detail in our co-pending U.S. application Ser. No. 09/340,776 claiming priority from GB Application No. 9930587.2, entitled A COMPUTER SYSTEM WITH DEBUG FACILITY FOR DEBUGGING A PROCESSOR CAPABLE OF PREDICATED EXECUTION.--

Please amend the fifth full paragraph on page 15, lines 15-17 as shown below.

-- The DG field indicates that a guard value has been modified and is used to ~~synchronise~~ synchronize guard values between the address unit pipelines and the data unit pipelines.--

Please amend the fourth full paragraph on page 16, lines 14-30 as shown below.

-- Thus, it can be seen that there are two fields in the microinstruction which allow guard value resolutions to be carried out, the SGLS and SGDU fields. As two microinstructions can be received by the data unit pipelines DU.sub.0, DU.sub.1 simultaneously, it is possible to convey four guards to be resolved to the data unit pipelines in these two microinstructions. This is useful because it allows all of the possible guards in a VLIW word to be transmitted to the data unit pipelines in the same cycle and thus for the guard values to be resolved without undue delay. This is dealt with in the manner shown in FIG. 8. The data unit execution pipelines DU.sub.0, DU.sub.1 contain circuitry for reading the SGDU and SGLS fields in conjunction with the relevant OCE bits to access the guard register file 30 and generate the Commit signal 52 accordingly. The execution pipelines DU.sub.0, DU.sub.1 deal with the fields in a predetermined order, i.e. DU.sub.0sgls; DU.sub.1sgls; DU.sub.0sgdu; DU.sub.1sgdu to allow ~~synchronisation~~ synchronization of the commit signals with respective PCs.--

Please amend the second paragraph on page 17, lines 8-27, as shown below.

-- For the sake of completeness, the mechanism used in the ~~synchronisation~~ synchronization unit 38 will be briefly explained with reference to FIG. 9. Instructions are output by the dispatch stage 24 which supplies to a program count FIFO (first in first out buffer) 120 an indication of the program count and an indication if the instruction is a load or store instruction. As the instruction passes through the pipeline stages of the data unit DU the guard value is resolved by hardware 101, 103 provided in the data unit for the normal execution of instructions in the data unit and is not additional hardware for use solely by the debugging operation. In this example the resolution is shown as occurring at stage e2 in the pipeline and the commit signal indicating whether the guard value is resolved as true or false is supplied to a commit FIFO 121. When a load/store instruction is executed in the pipeline within the address unit AU a signal is sent to a load store sent FIFO 122 to indicate whether or not the load/store has been dispatched by the address unit to a data memory controller (not shown). FIFO 120 receives its signals on line 42 of FIG. 2. FIFO 121 receives its signals on line 52 of FIG. 2.--

Please amend the paragraph beginning on page 17, line 28 through page 13, line 14, as shown below.

The timing of the ~~synchronisation~~ synchronization system ~~104~~ 38 will be explained with reference to FIG. 10. The cycles of operation of instruction fetches, execution pipelines and memory accesses are controlled by clock cycles with a clock signal as shown at ~~230~~ 130 in FIG. ~~7~~ 10. The figure illustrates ~~four~~ seven successive clock cycles and in this example the program count of the instruction dispatch by dispatch stage 24 occurs in cycle 2 as shown in the program count line 131. The commit signal is sent out in cycle 4 as shown in line 132. The load/store signal from the address unit is provided in cycle 5 as shown in line 133. It will be appreciated that the signal on line 131 was fed into FIFO 120. The signal on line 132 was fed into FIFO 121. The signal on line 133 was fed into FIFO 122. Each of the FIFOs 120,121 and 122 operate on ~~synchronised~~ synchronized clock cycles from the clock signal shown in FIG. 10. Each of the FIFOs 120-122 is then read in clock cycle 6 as shown by lines 136, 137 and 138 in FIG. 10. The result of reading each of those FIFOs on the same clock cycle 9 will indicate correlation between a commit signals and any of the events watched on lines 131-133. The emulator can therefore

through use of the ~~synchronisation~~ synchronization unit 38 establish the program count which was associated with a committed instruction and one which gave rise to a PC watch.